Key Performance Indicators of the Reference 6TiSCH Implementation in Internet-of-Things Scenarios

Mališa Vučinić, Tengfei Chang, Božidar Škrbić, Enis Kočan, Milica Pejanović-Djurišić, Thomas Watteyne

Abstract

Tens of thousands of wireless industrial monitoring deployments exist today, logging more than 18 billion operating hours. These solutions have been around for over a decade and are based on standards such as WirelessHART and ISA100.11a to provide performance guarantees to the applications. The new trend in industry deployments is the convergence of operational and information technologies happening through the Industrial Internet of Things (IIoT) paradigm. The challenge is to bridge the performance of these well-proven industrial standards with the interoperability of IP-based systems. The Internet Engineering Task Force (IETF), the organization behind most of the technical solutions of the Internet, has produced a set of specifications with this requirement in mind. The output of this effort is the 6TiSCH protocol stack based on open standards, such as those that have played a key role in the Internet's ubiquitous adoption. The standardization of 6TiSCH is done. The state-of-the-art research work focus is on important, but niche, optimizations and performance evaluations of the 6TiSCH stack. This paper takes a different approach – it evaluates the performance of the standards-compliant 6TiSCH solution from the end user point of view. It does so on two experimental testbeds, in typical IoT test scenarios based on a well-defined experimentation methodology. We provide a set of Key Performance Indicators (KPIs) useful for the end user to decide whether the 6TiSCH technology is a good fit performance-wise for a particular use case. We demonstrate reliability of a vanilla open-source implementation of 6TiSCH above 99.99%, upstream latency on the order of a second and radio duty cycle well below 1%.

Internet of Things, 6TiSCH, Experimentation, Testbed Repeatability, Reproducibility.

Key Performance Indicators of the Reference 6TiSCH Implementation in Internet-of-Things Scenarios

Mališa Vučinić[1], Tengfei Chang[1], Božidar Škrbić[2], Enis Kočan[2], Milica Pejanović-Djurišić[2], and Thomas Watteyne[1]

[1]Inria, 2 rue Simone Iff, 75012 Paris, France (e-mail: firstname.lastname@inria.fr)
[2]Faculty of Electrical Engineering, University of Montenegro, Bulevar Džordža Vašingtona bb (e-mail: firstname.lastname@ucg.ac.me)

1 Introduction

The Industrial Internet of Things (IIoT) introduces the convergence of operational and information technologies in the industry deployments. It facilitates their integration with novel web-based systems through the usage of interoperable solutions. The de-facto wireless communication technology in industrial applications is Timeslotted Channel Hopping (TSCH), used for more than a decade in standards such as WirelessHART and ISA100.11a. Through the work of the Internet Engineering Task Force (IETF) and its 6TiSCH working group, TSCH technology is now ready to be used in IPv6 networks. The result of this effort that spanned several years and a mix of academic and industrial participants is the 6TiSCH protocol stack. The 6TiSCH stack bridges the performance of existing industrial standards while benefiting from the Internet's IPv6 interoperability. The stack is based on open standards, such as those that have played a key role in the Internet's ubiquitous adoption. The goal of this paper is to define Key Performance Indicators (KPIs) of the 6TiSCH stack, a methodology for their collection, and to present the results of an extensive experimentation campaign using a reference 6TiSCH implementation.

The 6TiSCH protocol stack is based on a modular architecture. A key component influencing the performance of the stack is the "Scheduling Function" (SF). The 6TiSCH working group standardized one example of a scheduling function called Minimal Scheduling Function (MSF) [1] that is suited for best-effort traffic. A wide variety of scheduling functions have been proposed in the academic literature [2–6], each tailored to different application requirements.

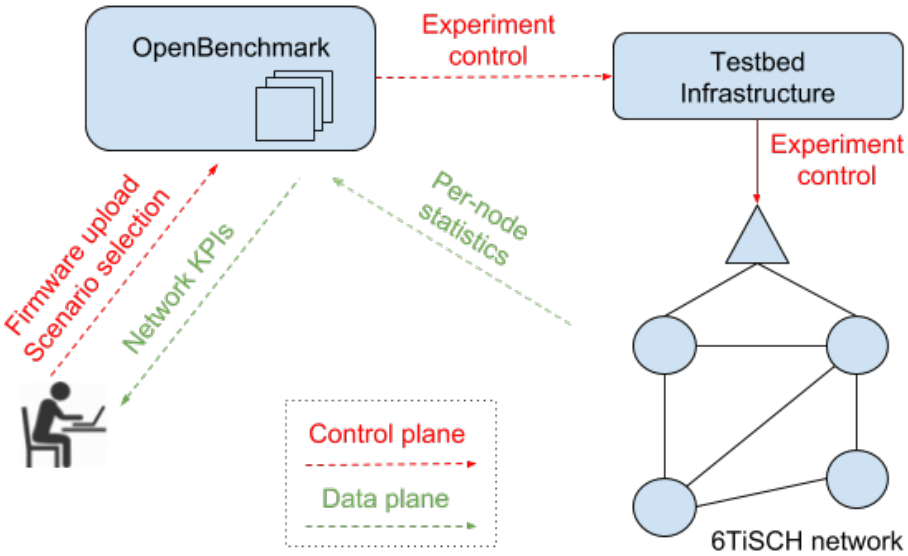

Figure 1: Overview of the OpenBenchmark functionality.

With many SFs available, how can one compare the performance in the context of different application requirements?

While there are many academic papers published on 6TiSCH, they typically discuss niche optimizations and their related performance improvements. While often very thorough, such evaluations fail to give a high-level view of the performance of the technology. The end users, e.g. product designers, are then left with a scattered view before deciding on whether to use a given technology. It is hard to find unbiased performance benchmark results for other IoT technologies either, although there is a plethora of academic proposals and evaluations available. We therefore approach the problem of an unbiased performance evaluation of the 6TiSCH protocol stack, as it was standardized by the IETF. We do not propose new optimizations, but rather evaluate the standards-compliant solution. We produce the KPIs that an industrial user would expect before deciding whether a technology suits its requirements.

To achieve this, we design a novel software-based platform called `OpenBenchmark`[1], which uses a black-box approach to benchmarking a 6TiSCH implementation. The concept of the platform is that the user should not worry about network specifics, but rather obtain high-level KPIs of a 6TiSCH implementation. The black-box approach facilitates the use of the platform by users that are not experts in low-power networking and firmware design. The user uploads the 6TiSCH firmware image, selects the test scenario and launches the experiment

[1]The article is an extension of the paper [7] published in the INFOCOM 2019 CNERT workshop. This version complements with the produced KPIs through an extensive experimentation campaign performed using the OpenBenchmark platform.

(see Fig. 1). The platform takes care of testbed resource provisioning, firmware programming, data collection and processing, and presents the user with a set of KPIs.

In order for the benchmark to be valuable to industrial users, `OpenBenchmark` instruments the firmware in real time during the experiment to adhere to a given test scenario. Test scenarios are defined to capture real-life use cases of a technology and therefore test its applicability. Since the test environment, i.e. a testbed, often plays an important role in performance results, the platform allows the experiments to be executed on different testbeds. For the purpose of this paper, we evaluate the reference 6TiSCH implementation, the OpenWSN stack [8], in industrial monitoring and home automation scenarios, each on two different testbeds to give performance insights.

In both scenarios, we observed reliability above 99%, which depending on the test environment goes up to 99.99%. Latency observed was on the order of a second and the radio duty cycle is well below 1%. It is important to stress that these results come from a vanilla open-source implementation of 6TiSCH. As each implementation can take different choices when implementing the standard, the performance of the implementations is likely to vary. As a consequence, these results should not be generalized as "performance of 6TiSCH". They should rather be seen as an example of a baseline when a reference open-source implementation of 6TiSCH is used. Furthermore, if application requirements are known in advance, many enhancements are possible. However, such optimizations are out of scope of this work.

The contribution of this paper is threefold:

- We obtain performance datasets of a reference 6TiSCH implementation in two test scenarios on two different testbeds and publish them under open-data licence[2];
- We analyze the datasets and discuss KPIs of the reference 6TiSCH implementation in each case;
- We design and implement in open-source `OpenBenchmark` and enable the community to leverage it for further evaluations or comparisons[3]. `OpenBenchmark` was developed as part of the SODA project [9] at the University of Montenegro.

The remainder of the article is organized as follows. Section 2 summarizes the related work on the subject of 6TiSCH performance evaluation. Section 3 presents the design of `OpenBenchmark`. Section 4 details the obtained KPIs in both test scenarios. Section 5 concludes this article.

[2]Datasets are available at `https://zenodo.org/record/3472626`

[3]As an online addition to this article, the source code of `OpenBenchmark` is published under a BSD open-source license at `https://github.com/openwsn-berkeley/openbenchmark`

2 Related Work

The work on standardizing 6TiSCH is complete. Core documents [1,10–13] have been published or are in the process of becoming Request for Comments (RFCs). During the process, 6TiSCH has sparked the interest of different communities, including open-source implementation projects, standardization and research.

The reference 6TiSCH implementation used during ETSI testing events for interoperability is the OpenWSN stack [8]. The two other major IoT open-source projects, Contiki-NG [14] and RIOT [15], implement 6TiSCH. The 6TiSCH simulator [16] implements a Python-based discrete-event simulation tool focusing exclusively on 6TiSCH. Other tools have also been developed focusing on interoperability and conformance testing of 6TiSCH implementations [17].

The performance evaluation of 6TiSCH networks has been a subject of interest of many academic works. The SF is the major component influencing the performance of the stack as it constructs the communication schedule of the network. Therefore, it comes to no surprise that the majority of the work in the literature proposes new scheduling functions [18]. Examples are DeTAS [2], Morell *et al.* [3], ReSF [4], LLSF [5], TREE [6]. Other work focuses on optimizing the joining [19, 20], interplay with routing [21], co-existence [22], applications [23] to time-critical scenarios [24, 25].

Many of these works evaluate their proposals in realistic conditions on different testbeds. While often very thorough, in the majority of cases, each work benchmarks its particular proposal with no common methodology and scenario followed. One consequence of this practice is that it is hard for an industrial user to find a comprehensive evaluation useful from the *application requirements* point of view. Our article fills this gap, by defining and following a methodology to evaluate the 6TiSCH network in scenarios relevant to the applications.

3 OpenBenchmark Platform

`OpenBenchmark` automates the experimentation and network performance benchmarking on selected testbeds supporting Internet of Things devices compliant with the IEEE802.15.4 standard. `OpenBenchmark` instruments the execution of an experiment in real time, following the pre-defined test scenarios, and collects the data to calculate the network KPIs in a fully automated manner.

Test scenarios are generic and derived from industrial requirements. A test scenario is mapped to an executable logic that runs concurrently with the experiment in the testbed. `OpenBenchmark` sends commands to trigger the desired actions of the firmware: configure radio transmit power, trigger application packet. The commands are sent to the Network Gateway, which processes and translates them into the potentially proprietary format expected by the firmware Implementation Under Test (IUT). The Network Gateway may run at the testbed infrastructure and be physically connected to the serial port of IUTs, or run at `OpenBenchmark` premises and communicate with the IUTs over an emulated serial port. This emulated serial port is provided through the software

component of the companion OpenTestbed project [26], which transports the serial data over the MQTT protocol. `OpenBenchmark` provides the necessary integration and provisioning of the OpenTestbed software on supported testbeds, such that this complexity is hidden from the user. This allows the user to focus on the protocol aspects of the firmware, while the performance evaluation is entirely handled by `OpenBenchmark` through the Application Programming Interfaces (APIs) exposed by compliant firmware projects.

3.1 Token-based Benchmarking

The benchmarking process of `OpenBenchmark` is based on random tokens. `OpenBenchmark` sends commands to the System Under Test (SUT) in real time, instrumenting it so that a node in the 6TiSCH network initiates the sending of an application packet. The command contains a 5-byte token that is to be transferred over the network by the originator node. Fig. 2 illustrates the process of `OpenBenchmark`, instrumenting node E to send an application packet to node A with a random token 3424. The command is received by the SUT Gateway and translated to the format understandable by the 6TiSCH Implementation Under Test (IUT). Upon the reception of the command, node E prepares an application packet and includes the token 3424 in its payload. SUT generates an MQTT event `packetSent` that is handled by `OpenBenchmark`, communicating the time instant at which the packet was sent, as well as other information necessary to calculate the KPIs. The packet is then handled by the 6TiSCH network and upon reception at node A, a new MQTT event is generated: `packetReceived`. The pair of `packetSent` and `packetReceived` events allows to calculate the latency of the packet and the number of hops traversed per packet. The absence of the `packetReceived` event indicates to `OpenBenchmark` that the packet has been dropped in the network, which consequently impacts the reliability.

One deficiency of the proposed design is in non-deterministic network delays between `OpenBenchmark` and the SUT Gateway. Since the commands that trigger the sending of a packet in the network are sent in real time, non-deterministic network delays between `OpenBenchmark` and the SUT Gateway do influence the reproducibility of the platform. To overcome this challenge, it would be necessary to implement a timestamp-based approach, where `OpenBenchmark` would communicate the exact timestamp at which the SUT Gateway should trigger the sending of an application packet in the network. The implementation of such timestamp-based approach is part of our future work.

3.2 Software Architecture

The `OpenBenchmark` platform consists of the following components (see Fig. 3) [7]:

- Agent. A component running at the Network Gateway side, translating `OpenBenchmark` commands to the format that the IUT implements, and also converting performance data from the IUT to the format expected by `OpenBenchmark`. The Agent component acts as both MQTT publisher

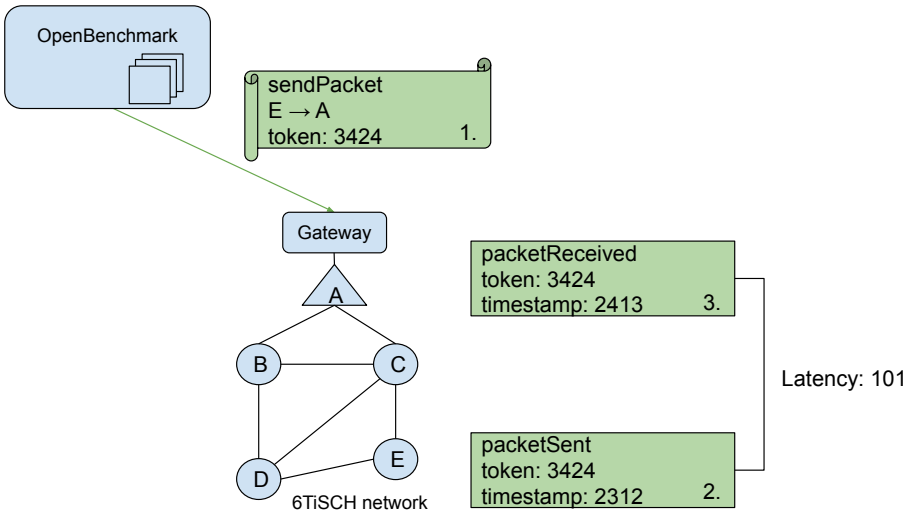

Figure 2: Token-based benchmarking illustration.

and subscriber. It publishes the events coming from the network towards the OpenBenchmark platform, needed to calculate the KPIs. It subscribes to the commands that the SUT should adhere to, coming from the Experiment Orchestrator.

- Experiment Controller. A component in charge of testbed node reservation, firmware flashing, and launching the necessary software components that run at testbed infrastructure side. The Experiment Controller bootstraps the testbed infrastructure by opening an SSH connection with each embedded computer in the testbed, and starting the execution of the MQTT publish/subscribe component in charge of emulating serial ports of the devices. The serial port emulation software (OpenTestbed) makes the testbed nodes appear to the Network Gateway as if they were physically connected. The Experiment Controller also starts the execution of the Network Gateway that can run either on OpenBenchmark or user premises and of the PHP backend that runs locally on OpenBenchmark premises.

- Experiment Orchestrator. A component in charge of orchestrating the SUT according to the selected test scenario. The Experiment Orchestrator interprets the test scenario files and instruments the experiment based on the interpreted data. The Experiment Orchestrator acts as an MQTT publisher and publishes to the broker the commands that the SUT needs to adhere to. The broker then dispatches these commands to the OpenBenchmark Agent based on the subscription to the common MQTT topics.

- Performance Event Handler. A component in charge of handling performance data events coming from the SUT. Based on these events, Performance Event Handler generates the experiment data sets and calculates the KPIs. The Performance Event Handler acts as an MQTT subscriber and receives events from the SUT, which it then uses to calculate the KPIs.
- Web server. A Laravel-based (PHP) backend and Vue.js-based frontend allowing the user to access the `OpenBenchmark` platform through a graphical interface. The backend serves as a bridge between the frontend and the rest of the `OpenBenchmark` components that are implemented in Python. The backend provides a RESTful API that enables the use of `OpenBenchmark` by 3^{rd} party applications.

3.3 Test Scenarios

The goal of an `OpenBenchmark` *test scenario* is to capture real-life use cases of a technology in order to benchmark its performance in a setting that is relevant to the end users: companies adopting the technology for their products and their customers. A test scenario also allows the experiment to be fully reproducible and the results easily and fairly comparable, desirable properties from a research point of view.

Each scenario describes the application traffic pattern and load, and the desirable coverage requirements in terms of number of IEEE802.15.4 hops. At a later stage, we plan on adding support for controllable interference generation. The description of a scenario is generic, with testbed-specific mappings.

3.3.1 Scenario Definition: Home Automation

Home automation systems typically consist of sensors monitoring some physical quantity, event sensors triggered by human action such as a button press, and different actuators. They are controlled by a central Control Unit (CU). The traffic consists of the mix of upstream and downstream traffic. The scenario has been derived from the requirements discussed in RFC5826 [27] and the emulated topology of a smart house discussed in Vučinić *et al.* [28]. Tables 1 and 2 summarize different logical roles a node in the network can have and the traffic pattern for each logical role.

3.3.2 Scenario Definition: Industrial Monitoring

Industrial monitoring systems can be generalized to consist of two types of sensors: 1) traditional monitoring sensors for temperature, pressure, fluid flow,...; 2) sensors that transmit large quantities of data, for example vibration monitors. They are controlled by a central Gateway. The traffic is typically upstream. Tables 3 and 4 summarize different logical roles a node in the network can have and the traffic pattern for each logical role. The scenario has been derived from the requirements discussed in RFC5673 [29].

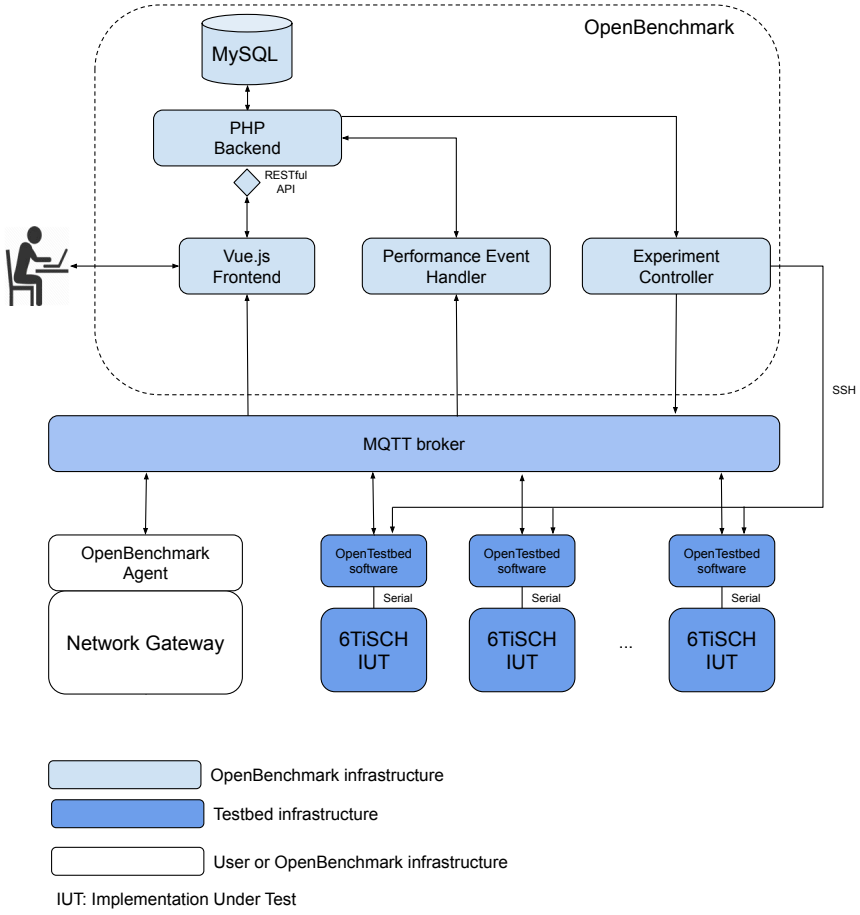

Figure 3: Software architecture of the OpenBenchmark platform. The System under Test (SUT) consists of the Network Gateway and firmware Implementations under Test (IUTs).

Table 1: Scenario "Home Automation": logical roles in the network.

Logical role	**Occurrence**	**Description**
Monitoring Sensor (MS)	49%	Sensor monitoring a physical value, e.g. temperature, humidity
Event Sensor (ES)	21%	Asynchronous event detection sensors, e.g. human presence
Actuator (A)	30%	Node performing some physical action, e.g. light dimmer, relay
Control Unit (CU)	1/network	Central unit controlling the automation system

Table 2: Scenario "Home Automation": traffic pattern.

Sender	**Dest.**	**Traffic pattern and load**	**Ack**
MS	CU	Periodic, uniformly in [3, 5] minutes	No
ES	CU	Poisson, mean of 10 packets/hour	Yes
A	CU	Periodic, uniformly in [3, 5] minutes	Yes
CU	multiple A	Poisson, mean of 10 5-packet bursts/hour	Yes

Table 3: Scenario "Industrial Monitoring": logical roles in the network.

Logical role	**Occurrence**	**Description**
Sensor (S)	90%	Traditional monitoring sensor: temperature, pressure, fluid flow, ...
Bursty Sensor (BS)	10%	Monitoring sensor transmitting large quantities of data: e.g. vibration monitor
Gateway (G)	1/network	Application gateway

Table 4: Scenario "Industrial Monitoring": traffic pattern.

Sender	**Dest.**	**Traffic pattern and load**	**Ack**
S	G	Periodic, uniformly in [10, 60] seconds	No
BS	G	Periodic, uniformly in [1, 60] minutes	No

3.4 Key Performance Indicators (KPIs)

In the following, we give a brief summary of implemented KPIs.

Reliability. Refers to the ratio between packets received and packets sent by the application. Therefore, this KPI indicates the end-to-end reliability. A packet may fail a transmission on a given link and later be re-transmitted. However, a failed packet transmission on a given link does not influence the end-to-end reliability if the packet eventually arrives at the destination. We present separately upstream reliability, referring to the packets destined for the Network Gateway, downstream reliability, referring to the packets originated by the Network Gateway and destined for one of the nodes in the 6TiSCH network, and P2P reliability, referring to the packets exchanged between a pair of 6TiSCH nodes.

Latency. Refers to the time interval between the instant packet is generated at the application layer of the sender, and the instant the packet is received by the application layer of the destination. We present separately upstream latency, downstream latency and P2P latency.

Radio Duty Cycle (RDC). Refers to the ratio between the cumulative time that the radio chip is powered and the measurement period. We present separately average duty cycle, minimal duty cycle and maximal duty cycle.

Network Formation Time. Refers to the initial phase when the network is forming. It is an important KPI from the installation point of view. The KPI refers to the end of the secure joining phase of the network.

3.5 Example Use Cases

We envision three main use cases of `OpenBenchmark`, with different target groups: IoT industry stakeholders, research community and firmware developers.

3.5.1 Referent Benchmark of an IoT Technology

Although there are many variants of IoT communication stacks (e.g. 6TiSCH, WirelessHART, ZigBee, ZigBee IP, Thread), it is quite challenging to point to a document that gives a fair and industry-relevant performance comparison among them. We designed `OpenBenchmark` to be used to tackle this challenge.

3.5.2 Research Proposal Benchmarking

The research community also benefits from `OpenBenchmark`. We hope to attract researchers to use our benchmarking service for the evaluations of their research proposals. `OpenBenchmark` facilitates the extraction of experiment data by hiding the unnecessary testbed complexity. Moreover, it also leads to the increased confidence in the results: `OpenBenchmark` is in its entirety open source and can be reviewed and improved by the community.

3.5.3 Continuous Delivery Benchmarking

Firmware always evolves. Updates to the standards, newly discovered security vulnerabilities in the code, new features, all require the firmware development community to constantly update the code base of different IoT open-source projects. The best practices of continuous integration testing are already in place for the popular repositories. However, unit and functional testing do not indicate whether a software patch introduces unwanted performance loopholes. *Does the proposed patch improve or degrade existing performance? In what conditions was the "existing performance" measured couple of years ago when we first merged that feature?* To answer such questions, `OpenBenchmark` is designed to provide a "continuous delivery benchmarking" service to firmware developers. We are working on integrating `OpenBenchmark` with the continuous integration procedures of the OpenWSN firmware project, the referent implementation of the 6TiSCH protocol stack. This allows the code maintainers to run automated nightly experiments and assess the performance of the latest patches, before their release.

4 Performance Evaluation

4.1 Methodology

The two test scenarios defined in 3.3 were instantiated and executed in order to collect data on two testbeds: Fed4Fire's w-iLab.t [30] in Ghent and Inria's OpenTestbed [26] in Paris. The data collection procedure was as following. Each scenario was instantiated for a total of 30 nodes in a generic setting including the root of the network. Then, a mapping was provided for each testbed, consisting of the testbed node_id to use, as well as the radio transmission power that is to be configured by `OpenBenchmark`. Listing 1 illustrates an example scenario instantiation and its mapping on w-iLab.t testbed.

The duration of each scenario execution was set to 3 hours and 30 minutes, with 30 minutes of allowance time for the network to form and stabilize before the benchmarking process would begin. We executed the two scenarios on w-iLab.t's Datacenter deployment using nodes `nuc28` to `nuc43`, each equipped with a pair of Zolertia Re-motes Rev. B. On OpenTestbed, we executed the scenarios using 30 OpenMote-B nodes in Building A of Inria-Paris deployment. In both cases, each scenario was executed using the same nodes, allowing us

Listing 1: An example JSON snippet showing a test scenario instantiation (left) and its mapping to the w-iLab.t testbed (right).

```
"identifier": "home-automation
    ",
"duration_min": 180,
"number_of_nodes": 30,
"nf_time_padding_min": 30,
"nodes": {
  "openbenchmark00": {
    "role": "control-unit",
    "area": null,
    "traffic_sending_points":
        [
    {
      "time_sec": 142.499,
      "payload_size": 10,
      "destination": "
          openbenchmark24",
      "packets_in_burst": 5,
      "confirmable": true
    },
    {
      "time_sec": 265.507,
      "payload_size": 10,
      "destination": "
          openbenchmark22",
      "packets_in_burst": 5,
      "confirmable": true
    },
    . . .

"openbenchmark00": {
  "node_id": "nuc10-1",
  "transmission_power_dbm": 0
},
"openbenchmark01": {
  "node_id": "nuc10-2",
  "transmission_power_dbm": 0
},
"openbenchmark02": {
  "node_id": "nuc10-3",
  "transmission_power_dbm": 0
},
"openbenchmark03": {
  "node_id": "nuc10-4",
  "transmission_power_dbm": 0
},
"openbenchmark04": {
  "node_id": "nuc10-5",
  "transmission_power_dbm": 0
},
"openbenchmark05": {
  "node_id": "nuc10-6",
  "transmission_power_dbm": 0
},
. . .
```

Table 5: Default parameters of the OpenWSN stack used for evaluation.

Parameter	Value
Application traffic	scenario-dependent
RPL DIO period	10 s
RPL DAO period	60 s
MSF max. number of cells	32
TSCH slotframe length	101 slots
TSCH slot duration	20 ms
TSCH EB transmission probability	0.1
TSCH max. number of retransmissions	15
Number of radio channels	16

to compare: 1) performance across scenarios; 2) performance across different testbeds and radio propagation conditions. We used the vanilla OpenWSN open-source project, with main parameters specified in Table 5.

We present KPIs in a tabular form, except for the network formation time that is presented as a Cumulative Distribution Function (CDF). For each KPI, we present the mean value, minimum, maximum and the 99th percentile ($P_{99\%}$, i.e. the value below which 99% of observations can be found) of at least 10 experiment runs. For example, if the discussed KPI is average latency, we present the mean, minimum, maximum and $P_{99\%}$ values over the experiment runs, where each measurement is the average latency in the network.

4.2 Network Formation Time

All the scenarios were executed using the same radio transmit power. As a consequence, due to the fixed physical topologies in the testbed, network formation time KPI is common across the scenarios. The plotted CDF (see Fig. 4) contains node join times across different scenarios.

From Fig. 4 we can see that it takes less than 20 minutes to form a 30-node network. This time is acceptable from the installation point of view as it does not require installers to spend an unreasonable amount of time on-site once the network is deployed. The time is consistent across the testbeds, which is interesting due to the fact that the deployments are quite different. w-iLab.t deployment used was the one in the Datacenter where all nodes have line-of-sight visibility of each other and OpenTestbed is deployed in a smart office setting across the floor of Inria-Paris building A. Even so, the network formed on w-iLab.t had a similar logical topology with the one formed on OpenTestbed in terms of the number of hops each packet would need to traverse. On w-iLab.t, the average number of hops was 2.5 while on OpenTestbed deployment in Paris, the average number of hops was 2.6.

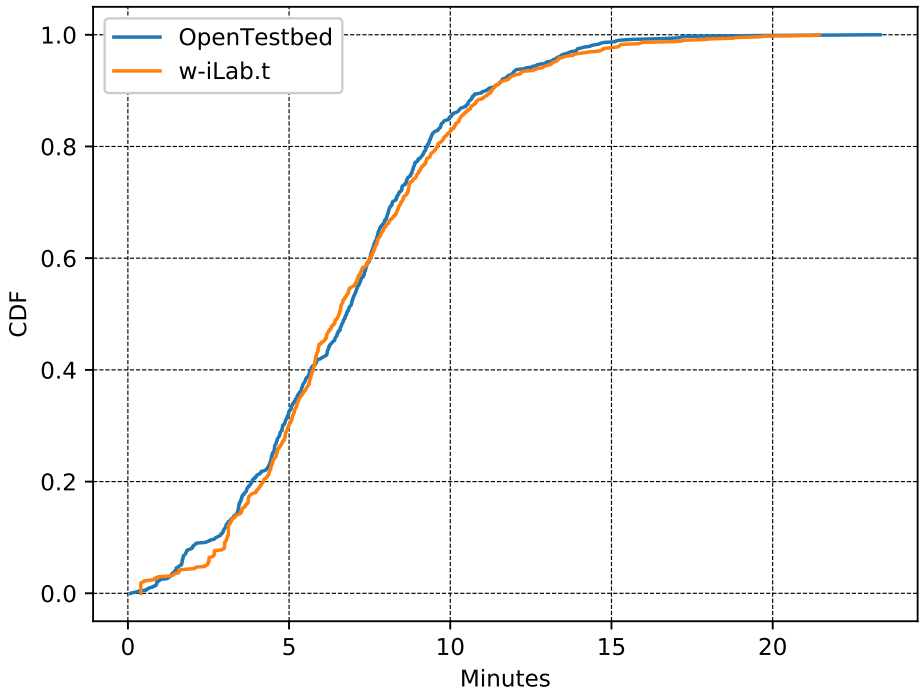

Figure 4: Network formation time CDF for different testbeds.

Table 6: Upstream reliability in "Industrial Monitoring" scenario.

Testbed	**Mean**	**Min**	**Max**	$P_{99\%}$
w-iLab.t	0.999954516	0.999795292	1	1
OpenTestbed	0.994725953	0.980388	0.999401854	0.999376399

Table 7: Reliability of bursty traffic in "Industrial Monitoring" scenario.

Testbed	**Mean**	**Min**	**Max**	$P_{99\%}$
w-iLab.t	1	1	1	1
OpenTestbed	0.993197279	0.952380952	1	1

4.3 Industrial Monitoring

Industrial monitoring scenario consists of exclusively upstream traffic with occasional bursts coming from bursty sensor node types. Each scenario execution run consisted of 10,861 packets being sent by different nodes in the network.

4.3.1 Reliability

Table 6 and Table 7 present the calculated reliability in the network for upstream and bursty traffic, respectively. During the experiments on w-iLab.t testbed in the Datacenter deployment, we observed four nines of reliability with some experiment runs without any losses.

The same scenario executed on OpenTestbed showed greater losses, equivalenting to 99.47% reliability of upstream communication. One explanation for this result is the radio interference present in the OpenTestbed deployment, causing higher losses on the radio channel.

We further studied the reliability of the traffic belonging to a burst and present the results in Table 7. We can see that during the experiment runs on w-iLab.t not a single packet belonging to a burst has been lost, which is not the case with the runs executed on the OpenTestbed deployment.

4.3.2 Latency

Table 8 presents the observed latency during the experiments, in TSCH slots and the equivalent in seconds for the slot length of 20ms used in the experiments. The interesting point to note here is that the results from the two testbeds are quite similar. This is a consequence of the logical network topologies built, with average hop distance from the root in both cases being less than 3 hops.

We further studied the latency of packets belonging to a burst and present the results in Table 9. We can see that the average latency of packets belonging to a burst is higher by a factor of 3 due to the queuing in nodes' buffers. Interestingly, the observation of similar latencies on two testbeds does not hold in the case of bursty traffic.

Table 8: Upstream latency in "Industrial Monitoring" scenario.

Testbed	**Mean**		**Min**		**Max**		$P_{99\%}$	
	Slots	Sec.	Slots	Sec.	Slots	Sec.	Slots	Sec.
w-iLab.t	179.93	3.60	142.12	2.84	221.76	4.44	220.18	4.40
OpenTestbed	179.34	3.59	150.11	3.00	200.94	4.02	200.85	4.02

Table 9: Latency of bursty traffic in "Industrial Monitoring" scenario.

Testbed	**Mean**		**Min**		**Max**		$P_{99\%}$	
	Slots	Sec.	Slots	Sec.	Slots	Sec.	Slots	Sec.
w-iLab.t	619.54	12.39	506.47	10.13	761.78	15.24	757.36	15.15
OpenTestbed	476.65	9.53	387.14	7.74	598.24	11.96	595.72	11.91

4.3.3 Radio Duty Cycle

Radio duty cycle is an important KPI from the energy consumption point of view as the radio transceiver typically accounts for the majority of current drawn on an IoT device. We present average duty cycle for the network in Table 10 and best case and worst case observations in Table 11 / Table 12, respectively. Best case, resp. worst case, refers to the lowest, resp. highest, observed duty cycle in a run.

From Table 11, we can see that the best-case result is quite consistent across the two testbeds and amounts to approximately 0.5%. The worst-case duty cycle in the network (see Table 12) is around 1.8% for the network formed on w-iLab.t and around 3.2% for the network formed on OpenTestbed. This is a consequence of the logical topology of the networks formed, as nodes closer to the root have more data to forward than the leaf nodes in the network. At 5 mA current drawn from the radio, a figure typical for state-of-the-art radio transceivers, this results in the average current draw from the radio at about 90uA on w-iLab.t and 160uA on OpenTestbed. To put this number into context, consider that a typical AA battery holds 2200mAh, so a worst-case node would approximately have 2.8 years of lifetime on a pair of AA batteries on w-iLab.t and 1.6 years on OpenTestbed, disregarding the microcontroller and sensor consumption. For comparison, the best-case node at the radio duty cycle of 0.5%, would need over 10 years before depleting a pair of AA batteries.

Table 10: Radio duty cycle (%) for the network in "Industrial Monitoring" scenario.

Testbed	**Mean**	**Min**	**Max**	$P_{99\%}$
w-iLab.t	0.731962963	0.709333333	0.753	0.752706667
OpenTestbed	0.855952381	0.805333333	1.076	1.06228

Table 11: Best case (minimum) radio duty cycle (%) for the network in "Industrial Monitoring" scenario.

Testbed	**Mean**	**Min**	**Max**	$P_{99\%}$
w-iLab.t	0.495555556	0.49	0.51	0.5092
OpenTestbed	0.522857143	0.5	0.54	0.5394

Table 12: Worst case (maximum) radio duty cycle (%) for the network in "Industrial Monitoring" scenario.

Testbed	**Mean**	**Min**	**Max**	$P_{99\%}$
w-iLab.t	1.793333333	1.62	2.11	2.0964
OpenTestbed	3.281428571	2.05	9.26	8.8478

4.4 Home Automation

Home automation scenario consists of a mix of upstream and downstream traffic. Downstream traffic consists of bursts as well as the application-layer acknowledgment packets. Each scenario run lasting 3 hours and 30 minutes consisted of 1272 packets being sent by different nodes in the network.

4.4.1 Reliability

Observed reliability of upstream traffic in the home-automation scenario is presented in Table 13. We can see that the average observed on w-iLab.t testbed is around 99,7%, while the same KPI observed on OpenTestbed deployment is 98,05%. We attribute this difference to the radio interference and different propagation conditions on the two testbeds.

For the case of downstream bursts, reliability is presented in Table 14. In both cases, downstream burst reliability is around 97%. The losses are attributed to the queue overflows due to the bursty nature of the traffic and the slow link capacity adaptation algorithm.

Table 15 presents the observed latency of upstream traffic. We observed average latency of 3.5 seconds on w-iLab.t and 4.8 seconds on OpenTestbed. Higher latency on OpenTestbed is partly the result of the deeper networks formed during the home automation scenario runs, where each packet traversed on average 2.86 hops, while on w-iLab.t each packet traversed on average 2.62 hops.

Table 13: Upstream reliability in "Home Automation" scenario.

Testbed	**Mean**	**Min**	**Max**	$P_{99\%}$
w-iLab.t	0.997097123	0.995515695	0.999188312	0.99911248
OpenTestbed	0.980507976	0.878919861	0.99918897	0.999188851

Table 14: Reliability of downstream bursty traffic in "Home Automation" scenario.

Testbed	**Mean**	**Min**	**Max**	$P_{99\%}$
w-iLab.t	0.971791888	0.968503937	0.976923077	0.976874016
OpenTestbed	0.968386473	0.93442623	0.984251969	0.983608223

Table 15: Upstream latency in "Home Automation" scenario.

Testbed	**Mean**		**Min**		**Max**		$P_{99\%}$	
	Slots	Sec.	Slots	Sec.	Slots	Sec.	Slots	Sec.
w-iLab.t	173.32	3.47	139.80	2.80	226.06	4.52	225.51	4.51
OpenTestbed	239.36	4.79	163.69	3.27	346.01	6.92	339.37	6.79

Table 16 presents the latency results for downstream bursty traffic. The observed latency for packets within a burst was 8.7 seconds on w-iLab.t while it was 12.2 seconds on OpenTestbed. It is important to note here that this result could be improved with the usage of shorter slots, as the default slot length in IEEE802.15.4 TSCH is 10ms, instead of 20ms used within the OpenWSN reference image. Indeed, using 10ms slots would halve the absolute latency in seconds.

Finally, Table 17 presents the downstream latency for non-bursty downstream traffic. Observed latency in case of w-iLab.t testbed was 3.96 seconds while on OpenTestbed it was 5.2 seconds.

4.4.2 Radio Duty Cycle

Table 18, Table 19, Table 20 present the observed results of radio duty cycle in the network while application traffic pattern is following the home automation scenario. Compared to the industrial monitoring scenario where the traffic load is higher, we can see that the duty cycle results are even better in the home automation case. The worst-case duty cycle in the network for home-automation was observed at 1.49% for w-iLab.t, and 1.68% for OpenTestbed.

Table 16: Latency of bursty traffic in "Home Automation" scenario.

Testbed	**Mean**		**Min**		**Max**		$P_{99\%}$	
	Slots	Sec.	Slots	Sec.	Slots	Sec.	Slots	Sec.
w-iLab.t	436.23	8.72	366.03	7.32	713.79	14.28	689.39	13.79
OpenTestbed	611.31	12.23	447.98	8.96	768.25	15.37	765.09	15.30

Table 17: Latency of downstream traffic in "Home Automation" scenario.

Testbed	**Mean**		**Min**		**Max**		$P_{99\%}$	
	Slots	Sec.	Slots	Sec.	Slots	Sec.	Slots	Sec.
w-iLab.t	198.21	3.96	148.51	2.97	282.03	5.64	278.07	5.56
OpenTestbed	261.22	5.22	192.62	3.85	335.91	6.72	332.36	6.65

Table 18: Radio duty cycle (%) for the network in "Home Automation" scenario.

Testbed	**Mean**	**Min**	**Max**	$P_{99\%}$
w-iLab.t	0.6748	0.665333333	0.695	0.69383
OpenTestbed	0.7163	0.703666667	0.736333333	0.735133333

Table 19: Best case (minimum) radio duty cycle (%) in "Home Automation" scenario.

Testbed	**Mean**	**Min**	**Max**	$P_{99\%}$
w-iLab.t	0.489	0.48	0.49	0.49
OpenTestbed	0.506	0.48	0.51	0.51

Table 20: Worst case (maximum) radio duty cycle (%) in "Home Automation" scenario.

Testbed	**Mean**	**Min**	**Max**	$P_{99\%}$
w-iLab.t	1.491	1.23	1.66	1.66
OpenTestbed	1.682	1.32	2.16	2.1321

5 Conclusion

The article presents the design of a benchmarking platform for IoT use cases `OpenBenchmark` and the benchmarking results of the reference implementation of the 6TiSCH protocol stack, the OpenWSN open-source project. `OpenBenchmark` is designed with end users in mind; it abstracts network and firmware specifics from the user and as an output presents the user with a set of KPIs relevant from the industrial point of view. The platform is also useful for evaluating research proposals using a well-defined methodology and a common set of KPIs. The source code of `OpenBenchmark` is available in open source.

We used `OpenBenchmark` to evaluate the performance of the reference implementation of 6TiSCH in industrial monitoring and home automation test scenarios. Each scenario was executed in two different radio environments, Inria's OpenTestbed in Paris, France and w-iLab.t in Ghent, Belgium.

From the results presented in previous section, we draw here some key takeaway in respect to the applicability of 6TiSCH as a technology to different application domains. We could see in industrial monitoring scenario that the observed reliability was above 99%, with experimental runs regularly showing 100% reliability. We observe high reliability also in the home automation scenario where some traffic is generated according to the Poisson distribution, mimicking human actions. In both scenarios, the observed latency can be up to 12 s in bursty traffic scenarios. This result can be easily improved by using shorter TSCH slot lengths or different scheduling approaches specifically for interactive applications. In both scenarios, the observed radio duty cycle below 1%, attesting of the low-power nature of the 6TiSCH technology. While the battery lifetime is a board-level aspect with the attached sensors and the microcontroller also playing an important role, we could see that the consumption of the radio transceiver was negligible and allowing, alone, for a battery lifetime on a pair of AA batteries over 10 years.

Finally, it is important to note here that we used a vanilla version of the OpenWSN firmware image of 6TiSCH without any specific optimizations to a specific use case. Knowing the application traffic patterns and load in advance, it is straightforward to further tune the solution to find a different trade-off between latency and energy consumption for example.

As part of our future work, we plan on extending `OpenBenchmark` to other IoT technologies and platforms. Indeed, it would be interesting to compare results between different IoT technologies for common application traffic patterns, as defined by our test scenarios.

References

[1] T. Chang, M. Vučinić, X. Vilajosana, and D. Dujovne, *6TiSCH Minimal Scheduling Function (MSF)*, Internet Engineering Task Force Std. draft-ietf-6tisch-msf (work in progress), December 2019.

[2] N. Accettura, M. R. Palattella, G. Boggia, L. A. Grieco, and M. Dohler, "Decentralized Traffic Aware Scheduling for Multi-Hop Low Power Lossy Networks in the Internet of Things," in *2013 IEEE 14th International Symposium and Workshops on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, June 2013, p. 1–6.

[3] A. Morell, X. Vilajosana, J. L. Vicario, and T. Watteyne, "Label switching over IEEE802. 15.4 e networks," *Transactions on Emerging Telecommunications Technologies, Wiley Online Library*, vol. 24, no. 5, pp. 458–475, 2013.

[4] G. Daneels, B. Spinnewyn, S. Latré, and J. Famaey, "ReSF: Recurrent low-latency scheduling in IEEE 802.15. 4e TSCH networks," *Ad Hoc Networks, Elsevier*, vol. 69, pp. 100–114, Feb. 2018.

[5] T. Chang, T. Watteyne, Q. Wang, and X. Vilajosana, "LLSF: Low Latency Scheduling Function for 6TiSCH Networks," in *International Conference on Distributed Computing in Sensor Systems (DCOSS)*. IEEE, May 2016, pp. 93–95.

[6] T. van der Lee, G. Exarchakos, and S. H. de Groot, "Swarm-based energy efficient scheduling for wireless sensor networks," in *2019 IEEE Conference on Standards for Communications and Networking (CSCN)*. IEEE, 2019, pp. 1–6.

[7] M. Vučinić, B. Škrbić, E. Kočan, M. Pejanović-Djurišić, and T. Watteyne, "OpenBenchmark: Repeatable and Reproducible Internet of Things Experimentation on Testbeds," in *IEEE INFOCOM, CNERT workshop*. IEEE, 2019.

[8] T. Watteyne, X. Vilajosana, B. Kerkez, F. Chraim, K. Weekly, Q. Wang, S. Glaser, and K. Pister, "OpenWSN: a standards-based low-power wireless development environment," *Transactions on Emerging Telecommunications Technologies*, vol. 23, no. 5, pp. 480–493, 2012.

[9] M. Vučinić, M. Pejanović-Djurišić, and T. Watteyne, "SODA: 6TiSCH Open Data Action," in *2018 IEEE Workshop on Benchmarking Cyber-Physical Networks and Systems (CPSBench)*. IEEE, 2018, pp. 42–46.

[10] X. Vilajosana, K. S. Pister, and T. Watteyne, *Minimal IPv6 over the TSCH Mode of IEEE 802.15.4e (6TiSCH) Configuration*, Internet Engineering Task Force Std. RFC8180, May 2017.

[11] Q. Wang, X. Vilajosana, and T. Watteyne, *6TiSCH Operation Sublayer (6top) Protocol (6P)*, Internet Engineering Task Force Std. RFC8480, November 2018.

[12] M. Vučinić, J. Simon, K. S. Pister, and M. Richardson, *Constrained Join Protocol (CoJP) for 6TiSCH*, Internet Engineering Task Force Std. draft-ietf-6tisch-minimal-security-15 (work in progress), December 2019.

[13] P. Thubert, *An Architecture for IPv6 over the TSCH mode of IEEE 802.15.4*, IETF Std. draft-ietf-6tisch-architecture-20 [work-in-progress], 2019.

[14] S. Duquennoy, A. Elsts, A. Nahas, and G. Oikonomou, "TSCH and 6TiSCH for Contiki: Challenges, Design and Evaluation," in *International Conference on Distributed Computing in Sensor Systems (DCOSS)*, Ottawa, Canada, June 2017, pp. 1–8.

[15] E. Baccelli, O. Hahm, M. Günes, M. Wählisch, and T. Schmidt, "RIOT OS: Towards an OS for the Internet of Things," in *IEEE International Conference on Computer Communications (INFOCOM)*, Turin, Italy, April 2013.

[16] E. Municio, G. Daneels, M. Vučinić, S. Latre, J. Famaey, Y. Tanaka, K. Brun-Laguna, K. Muraoka, X. Vilajosana, and T. Watteyne, "Simulating 6TiSCH networks," *Transactions on Emerging Telecommunications Technologies, Wiley*, vol. 30, no. 3, Mar. 2019.

[17] S. Ziegler, S. Fdida, T. Watteyne, and C. Viho, "F-Interop - Online Conformance, Interoperability and Performance Tests for the IoT," in *Conference on Interoperability in IoT (InterIoT)*, Paris, France, October 2016.

[18] A. Elsts, S. Kim, H.-S. Kim, and C. Kim, "An Empirical Survey of Autonomous Scheduling Methods for TSCH," *IEEE Access*, 2020.

[19] B. Al Nahas, S. Duquennoy, and O. Landsiedel, "Network Bootstrapping and Leader Election in Low-power Wireless Networks," in *ACM SenSys*. ACM, 2017.

[20] M. Vučinić, T. Watteyne, and X. Vilajosana, "Broadcasting Strategies in 6TiSCH Networks," *Wiley Internet Technology Letters*, November 2017.

[21] F. Righetti, C. Vallati, G. Anastasi, and S. Das, "Performance Evaluation the 6top Protocol and Analysis of its Interplay with Routing," in *Smart Computing (SMARTCOMP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 1–6.

[22] S. B. Yaala, F. Théoleyre, and R. Bouallegue, "Cooperative resynchronization to improve the reliability of colocated IEEE 802.15. 4-TSCH networks in dense deployments," *Ad Hoc Networks*, vol. 64, pp. 112–126, 2017.

[23] A. Karaagac, J. Haxhibeqiri, I. Moerman, and J. Hoebeke, "Time-critical communication in 6TiSCH networks," in *2018 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*. IEEE, 2018, pp. 161–166.

[24] L. Toka, B. Lajtha, É. Hosszu, B. Formanek, D. Géhberger, and J. Tapolcai, "A resource-aware and time-critical iot framework," in *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*. IEEE, 2017, pp. 1–9.

[25] P. Štefanič, M. Cigale, A. C. Jones, L. Knight, I. Taylor, C. Istrate, G. Suciu, A. Ulisses, V. Stankovski, S. Taherizadeh *et al.*, "SWITCH workbench: A novel approach for the development and deployment of time-critical microservice-based cloud-native applications," *Future Generation Computer Systems*, pp. 197–212, 2019.

[26] J. Muñoz, F. Rincon, T. Chang, X. Vilajosana, B. Vermeulen, T. Walcarius, W. Van de Meerssche, and T. Watteyne, "OpenTestBed: Poor Man's IoT Testbed," in *IEEE INFOCOM, CNERT workshop*. IEEE, 2019.

[27] A. Brandt, J. Buron, and G. Porcu, *Home Automation Routing Requirements in Low-Power and Lossy Networks*, Internet Engineering Task Force Std. RFC5826, April 2010.

[28] M. Vučinić, B. Tourancheau, and A. Duda, "Performance comparison of the RPL and LOADng routing protocols in a home automation scenario," in *Wireless Communications and Networking Conference (WCNC), 2013 IEEE*. IEEE, 2013, pp. 1974–1979.

[29] K. Pister, P. Thubert, S. Dwars, and T. Phinney, *Industrial Routing Requirements in Low-Power and Lossy Networks*, Internet Engineering Task Force Std. RFC5673, October 2009.

[30] "Wireless Testlab and OfficeLab," https://doc.ilabt.imec.be/ilabt/wilab/, accessed December, 13th 2019.

[31] T. Qiu, B. Li, W. Qu, E. Ahmed, and X. Wang, "TOSG: A topology optimization scheme with global small world for industrial heterogeneous Internet of Things," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3174–3184, 2018.

[32] N. Chen, T. Qiu, X. Zhou, K. Li, and M. Atiquzzaman, "An Intelligent Robust Networking Mechanism for the Internet of Things," *IEEE Communications Magazine*, vol. 57, no. 11, pp. 91–95, 2019.

Mališa Vučinić is a Research Scientist with the EVA team of Inria in Paris, France. He received the Engineering degree from the University of Montenegro in 2010, the joint Master's (Hons.) degree from the Politecnico di Torino and the Grenoble Institute of Technology in 2012, and the Ph.D. degree from the Grenoble Alps University in 2015. From 2012 to 2015, he was a Research Engineer with STMicroelectronics, and was a Visiting Scholar with the University of California at Berkeley in 2015. Mališa is active in the IETF where he co-chairs the LAKE working group on security and leads the security standardization work in 6TiSCH. He is a core developer of the OpenWSN project, the reference 6TiSCH implementation, and a co-lead of the 6TiSCH simulator. His current research interests include the intersection of communication security and performance analysis in Internet-of-Things scenarios.

Tengfei Chang is a Postdoc Research Engineer at Inria-EVA, Paris. He obtained his Ph.D. degree in Computer System Architecture at 2017 from University of Science and Technology, Beijing. At 2014, he was visiting at the University of California, Berkeley as visiting scholar. From November 2015 to October 2017, he joined Inria-EVA team as a Pre-Postdoc Research Engineer, leading the project of OpenWSN, which is an Open Source project founded by UC Berkeley. At 2017, he joined the F-Interop project as a Postdoc Research Engineer, which is an H2020 European research project. He is also one of the main implementors of IETF 6TiSCH standard protocol stack. He has worked as technical support for 6TiSCH interoperability plugtest. He has huge interests on Wireless Sensor and Actuator Network, Swarm Robotic and any Embedded System design.

Božidar Škrbić received B.Sc degree in 2015 at University of Montenegro, Faculty of Electrical Engineering in Podgorica, at the department of Electronics, Telecommunications and Computer sciences, earning Best student award. In 2016 he finished postgraduate studies and earned Spec.Sci degree in Computer Sciences. As a software engineer he was part of BIO-ICT Centre of Excellence at the Faculty of Electrical Engineering in Podgorica, from October 2016 to September 2018, participating in the development of several major prototypes and software solutions. From September 2018 until September 2019 he was a software developer at SODA, a project of University of Montenegro funded by the H2020 Fed4FIRE+ consortium. Currently, he is a student of master studies at University of Montenegro, Faculty of Electrical Engineering, specializing in machine learning.

Enis Kočan is an associate professor at Faculty of Electrical Engineering, University of Montenegro. He received MSc and PhD degrees in Telecommunications in 2005 and 2011, respectively, both from the University of Montenegro. Part of his PhD research Enis has conducted at Aristotle University of Thessaloniki. He has published more than 70 scientific papers in international journals and peer reviewed conferences. He is recipient of the Best Paper Award at the International conference on Wireless Personal Multimedia Communications (WPMC 2013), held in the frame of the Global Wireless Summit in 2013. His research areas include digital communications over fading channels, with particular emphasis on the OFDM based cooperative communications, solutions for 5G networks, IoT wireless communication solutions and techniques for exposure reduction in wireless communication systems.

Milica Pejanović-Djurišić is full professor in telecommunications at the University of Montenegro, Faculty of Electrical Engineering, Podgorica, Montenegro. She has published more than 200 scientific papers in peer-reviewed international and national journals and conference proceedings, being the author of four books and a number of book chapters. Her main research interests are: wireless communications, 5G wireless networks, wireless IoT, cooperative and energy efficient transmission techniques, ICT trends and applications, optimization of telecommunication development policy. Prof. Pejanović-Djurišić has considerable industry and operating experiences working as industry consultant and Telecom Montenegro Chairman of the Board. She has been in charge of wire-

less networks design and implementation in Montenegro and in the region of SE Europe. Prof. Pejanović-Djurišić has been leading and coordinating many internationally and EU funded ICT projects and initiatives. She is a member of IEEE and IEICE, with a long engagement in the field of telecommunication regulation and standardization. In addition to work on national and regional levels, she has participated, in cooperation with ITU, in a number of global missions and activities related with regulation issues, development strategies and new technological solutions.

Thomas Watteyne is an insatiable enthusiast of low-power wireless mesh technologies. He holds a Research Director position at Inria in Paris, in the EVA research team, where he leads a team that designs, models and builds networking solutions based on a variety of Internet-of-Things (IoT) standards. Since 2013, he co-chairs the IETF 6TiSCH working group, which standardizes how to use IEEE802.15.4e TSCH in IPv6-enabled mesh networks, and is member of the IETF Internet-of-Things Directorate. Prior to that, Thomas was a postdoctoral research lead in Prof. Kristofer Pister's team at the University of California, Berkeley. Between 2005 and 2008, he was a research engineer at France Telecom, Orange Labs. He holds a PhD in Computer Science (2008), an MSc in Networking (2005) and an MEng in Telecommunications (2005) from INSA Lyon, France. He is Senior member of IEEE. He is fluent in 4 languages.